ARMY RESEARCH LABORATORY

General Architecture for Text Engineering (GATE) Developer for Entity Extraction: Overview for SYNCOIN

by Michelle Vanni and Andrew Neiderer

ARL-TR-7000 **July 2014**

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

Army Research Laboratory
Aberdeen Proving Ground, MD 21005-5067

ARL-TR-7000 **July 2014**

General Architecture for Text Engineering (GATE) Developer for Entity Extraction: Overview for SYNCOIN

Michelle Vanni and Andrew Neiderer
Computational and Information Sciences Directorate, ARL

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
July 2014	Final	November 2012–September 2013

4. TITLE AND SUBTITLE
General Architecture for Text Engineering (GATE) Developer for Entity Extraction: Overview for SYNCOIN

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Michelle Vanni and Andrew Neiderer

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army Research Laboratory
ATTN: RDRL-CII C
Aberdeen Proving Ground, MD 21005-5067

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TR-7000

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
The integrated development environment of the General Architecture for Text Engineering (GATE), or GATE Developer, is used to annotate entities in a text document consisting of messages in and around the Baghdad area (SYNCOIN data). Highlighting entities, such as person(s), location(s), and organization(s), may result in a more structured format for faster comprehension of the data. The application for entity determination is called a nearly-new information extraction, or ANNIE: a system of seven processing resources (PRs) in GATE. ANNIE is executed from the graphical user interface (GUI). Other PRs, such as those for machine learning, and the capability for user-defined applications are managed as a collection of reusable objects for language engineering (CREOLE); an icon for the CREOLE plug-in manager exists at the GUI as well.

15. SUBJECT TERMS
GATE Developer, SYNCOIN, ANNIE, gazetteer lists, JAPE rules, entity extraction, annotation toolkit

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Andrew Neiderer
Unclassified	Unclassified	Unclassified	UU	28	19b. TELEPHONE NUMBER (*Include area code*) 410-278-3203

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures **iv**

1. Introduction **1**

2. GATE Developer Graphical User Interface (GUI) **2**

2.1 Extraction and Guidelines 2

2.2 GATE Terminology 3

2.3 GATE Developer GUI....3

3. GATE Developer Customization **5**

3.1 GATE Interfaces....5

3.2 Adapting Resources....6

3.3 Ambiguity....7

3.4 JAPE Pattern Matching 7

3.5 Annotation Output....10

4. Machine Learning Resources **11**

5. Conclusions and Future Work **12**

6. Bibliography **14**

Appendix A. Running A Nearly New Information Extraction (ANNIE) Extraction on a File of 595 Concatenated SYNCOIN Messages **17**

Appendix B. Running A Nearly New Information Extraction (ANNIE) Extraction on a Corpus of SYNCOIN Messages **19**

Distribution List **22**

List of Figures

Figure 1. Interacting with GATE: Gate Developer, GATE Embedded. 1
Figure 2. GATE Developer GUI. 4
Figure 3. GUI and command line listings of ANNIE PRs. 6
Figure 4. A simple JAPE grammar rule. 9

1. Introduction

This report records knowledge gained at the U.S. Army Research Laboratory (ARL) on the use of the General Architecture for Text Engineering (GATE). Flexibility and ease of use were the principal factors influencing our choice of GATE in support of a Named Entity (NE) Extraction (NEE) task to be performed on the synthetic message dataset known as SYNCOIN.* The SYNCOIN dataset consists of varying types of messages contrived by military scenario developers to have been sent and received by individuals living in and around the Baghdad area.

As for GATE, its development started in 1995 at the University of Sheffield, United Kingdom, and has grown in complexity, robustness, and renown. GATE software consists of the GATE Developer graphical user interface (GUI) and the GATE Embedded applications programming interface (API)† (see figure 1). Its user community boasts support from world-class computational linguists and its Wiki site provides GATE training material for courses the University of Sheffield has been offering since 2009.‡

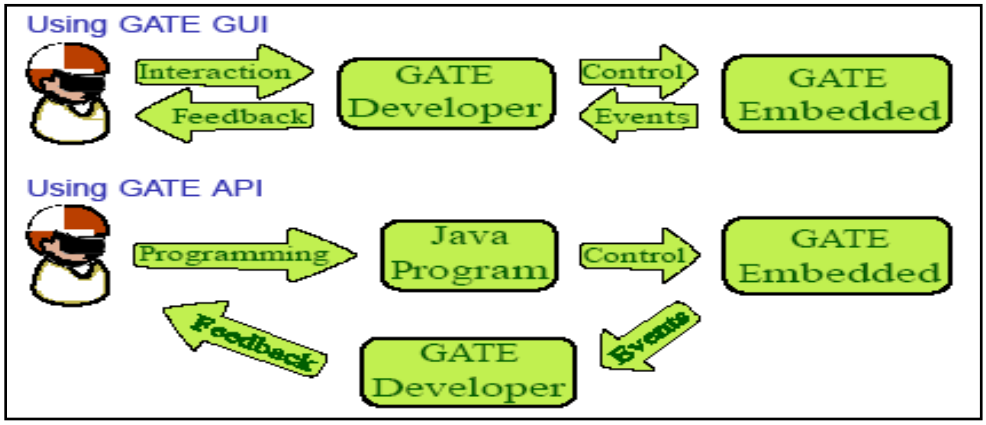


Figure 1. Interacting with GATE: Gate Developer, GATE Embedded.§

*The SYNCOIN corpus is described in Graham, J. L.; Hall, D. L.; Rimland, J. A Synthetic Dataset for Evaluating Hard and Soft Fusion Algorithms. Presented at the 14th International Conference on Information Fusion, Chicago, IL, 5–8 July 2011. http://speidigitallibrary.org/data/Conferences/SPIEP/62118/80620F_1.pdf (accessed February 2013).

†GATE is open-source software downloadable from http://gate.ac.uk/download/; email questions are fielded at gate-users@lists.sourceforge.net and archived at http://sourceforge.net/mailarchive/forum.php?forum_name=gate-users.

‡https://gate.ac.uk/wiki/. Note that the screenshots, examples and listings in figures 1–4 are taken from already published online sources, which are acknowledged in the footnotes.

§Source: The GATE Embedded API, Track 11, Module 5, Fifth GATE Training Course, slide 7 of 61, copyrighted by the University of Sheffield, June 2012; GATE Website. https://gate.ac.uk/sale/talks/gate-course-may-10/track-2/module-5-embedded/module-5-slides.pdf, accessed (15 July 2013).

The three-part layout of the GATE Developer interface, with its Panes for Resources, Display and Annotation, is the focus of section 2 of this report. In section 3, we detail grammar rules and lexical resources, such as collections of names and technical terminology as well as lists of closed class items often falling into part-of-speech (POS) categories of preposition, conjunction and pronoun. GATE resources for machine learning (ML) are described in section 4. In section 5, we conclude and discuss future efforts.

2. GATE Developer Graphical User Interface (GUI)

2.1 Extraction and Guidelines

While there are numerous capabilities housed in GATE, its featured functionality, and the one for which it is best known, is NEE, which constitutes a subset of the larger natural language processing (NLP) problem known as Information Extraction (IE). The intent of IE is to pull out from a text those well-defined tokens that match a specific definition of an information type or category such as, in the case of named entities, Person (PER), Location (LOC) and Organization (ORG) names. Highly qualified linguists perform this process manually to create ground truth data, by studying the principles set forth and exemplified in a project's annotation guidelines or coding manuals.[*]

SYNCOIN data experiments use ground truth annotations prepared according to a standard known as Simple Named Entity Guidelines[1] (SNEG). Based on the MUC-7 NE definitions,[2] this standard is also used in a nearly new information extraction (ANNIE) tutorial for PER, LOC and ORG NE recognition.[3]

The manually annotated ground truth data is used to train and test automatic IE engines. One such engine is GATE's. GATE orders ANNIE's several rule-based (RB) processors and sends their returns to subsequent routines.[†] For any given annotation category, RB IE engines compare candidate text strings, or the text surrounding them, against known list item strings or their abstract representations. The engines insert annotations corresponding to the category type around any matching string tokens found.

[*]Humans annotating text for ML use the term "coding manual" to refer to the highly precise category definitions that guide a project's analytical tagging of ground truth data for system training and evaluation.

[1]Linguistic Data Consortium Webpage. Simple Named Entity Guidelines for Less Commonly Taught Languages; v6.5; March 2006. http://projects.ldc.upenn.edu/LCTL/Specifications/SimpleNamedEntityGuidelinesV6.5.pdf (accessed December 2012).

[2]Chinchor, N. MUC-7 NE Task Definition, v3.5, 1997. http://www.itl.nist.gov/iaui/.../ne_task.html (accessed December 2012).

[3]Cunningham, H. Bontcheva, K. NE Recognition. http://gate.ac.uk/sale/talks/ne-tutorial.ppt, 09/08/2003 (accessed December 2012).

[†]The seven-stage GATE ANNIE pipeline is described in section 3. External IEs LingPipe and OpenNLP can be installed using the CREOLE (Collection of REusable Objects for Language Engineering) plug-in manager.

Developers of IE engines designed to handle large volumes of data avoid the characteristic brittleness of RB systems with probabilistic language models, built using ML algorithms (see section 4). Once trained on ground truth text, these models guide the ML IE engines to recognize text strings weighted heavily toward correspondence with a given annotation category and to insert annotations around those strings. Although potentially more reliable on unseen data than RB IE, ML IE requires a very large quantity of training data. The volume of data required increases naturally with the complexity and robustness of the annotation schemes. Also required are computing capacity, speed and power capable of training a model, and testing an engine with reasonable amounts of equipment and time.

Regardless of approach, however, the accuracy of automatic IE is measured by comparing engine placement of annotations against human placement of annotations on the same set-aside portion of ground truth text data.* Annotations create structure in text, which is valuable for information processing because it permits category-specific downstream processing. This may include, for one, interface displays with category-specific colors for ease of human content analysis and, for another, software designs with category-specific string handling for gains in system and application performance.

2.2 GATE Terminology

The GATE framework consists of two basic types of resources, processing resources (PRs) and language resources (LRs). GATE PRs are implementations of algorithms that take as input text files, i.e., LRs in GATE. A PR returns an annotated or otherwise processed text file, which is also an LR. The term "application," or "plug-in," is used to refer to a PR, or two or more PRs, arranged in a predetermined order to achieve a specific effect. GATE's ANNIE system is a well-known and widely used example of a GATE application, which can be adapted for use on particular types of data.

2.3 GATE Developer GUI

Figure 2 shows the layout of the GATE Developer 7.1 GUI. Horizontally displayed across the top is the (1) menu bar and the (2) icon bar just below it. For the project displayed, icons for frequently used actions in GATE Developer include (3) Restore Application from File, (4) Load ANNIE System, (5) New Language Resource, (6) New Processing Resource (PR), (7) New Application, (8) Data Stores, (9) Manage CREOLE Plug-Ins, and (10) Annotation Differences. Icons for actions 4–8 appear vertically in the Resources pane for every application, while those for 3, 9 and 10, appearing horizontally in the icon bar, are project-specific.

*Human programmers of the system engine and human annotators of the ground truth follow identical guidelines.

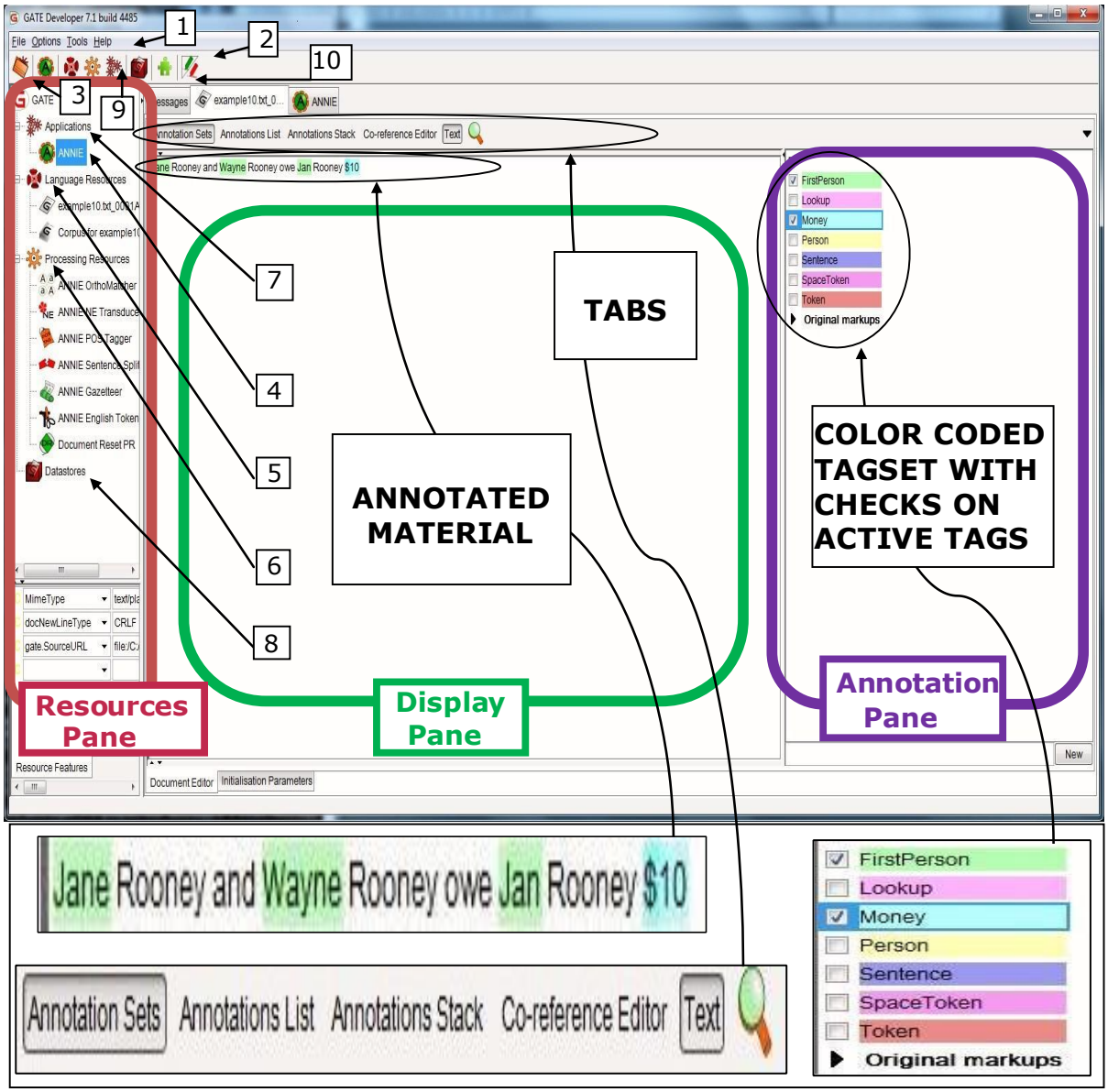


Figure 2. GATE Developer GUI.

A set of tabs are displayed, starting toward the center and proceeding from left to right, which consist of the following system features and can be accessed by selecting a tab: Annotation Sets, Annotations List, Annotations Stack, Co-Reference Editor, and the Text to be annotated. Note that both the annotation set displayed in the Annotation pane and the text sentence displayed in the Display pane correspond to the project's active tabs, Annotation Sets and Text. The intent here is to provide familiarization with the GATE Developer GUI for manipulation of actual content and resources. For a detailed discussion of each of the tabs, consult the GATE manual site.[4]

[4]GATE website. http://gate.ac.uk/sale/tao/ (accessed December 2012).

The example in figure 2 consists of the sentence, "Jane Rooney and Wayne Rooney owe Jan Rooney $10."* Note that strings 'Jane,' 'Wayne,' and 'Jan' have been annotated with tag 'FirstPerson,' and the string '$10' with tag 'Money,' as indicated by highlighting in green and blue, respectively.† When changes are saved, gazetteers or PRs that match list items to input strings for annotation with tags 'FirstPerson' and 'Money' will be updated.

Reference to the GUI is made often throughout this report, particularly in the appendices. GUI use for manual annotation and resource updating was detailed in previous paragraphs. Yet, appendix A, for example, describes a stepwise process to automatically annotate a single file of concatenated messages. For that, ANNIE automatic IE is run from the Resources pane on the Text in the Display pane. Annotation set(s) of information-category-defined annotation types are computed automatically by GATE. The sets and types can also be tailored to match features of a specific task, text genre, topic domain, formatting style, or combinations thereof, as described in section 3.

IE output displays as original input text, with GATE-computed annotation types displayed in the GUI's Annotation pane. Clicking on the checkbox to the left of a type causes its referring expressions to be highlighted with the appropriate—unique to its category—color within the text.

3. GATE Developer Customization

3.1 GATE Interfaces

As mentioned in section 2, GATE's GUI is versatile software, integral to GATE Developer, which permits viewing of input and output, manual annotation, and resource adaptation. The latter is only one of several GUI-accessible functionalities available for IE. When the PR functions are ordered into an application, such as ANNIE, the resultant RB entity-centric IE incorporates a Gazetteer-entry matching routine and a Java Annotation Pattern Engine (JAPE), as discussed in section 3.4. These resources can be created and edited without programming GATE Embedded, making the GUI an effective tool for quick text category mark-up by analysts. When category tokens are known in advance, lists can be added in "batch" mode from the command line. Regardless, at least one new list and one new grammar will effect domain capture in GATE.

ANNIE is GATE's flagship IE application with a pipeline consisting of the following ordered PRs: (1) Orthomatcher /Orthographic Co-Reference, (2) NE Transducer, (3) POS Tagger, (4)

*Thakker, D.; Osman, T.; Lakin, P. GATE JAPE Grammar Tutorial v. 1.0., 2009. GATE Website. https://gate.ac.uk/sale /thakker-jape-tutorial (accessed 11 March 2013).

†First names of PERson entities are annotated with the "FirstPerson" tag and expressions involving currency or other legal tender are annotated with the "Money" tag.

‡Dr. Paula Matuszek summarized each of these GATE PRs, or algorithms, in the 2012 text mining presentation at http://www.csc.villanova.edu/~matuszek/spring2012/GATEOverview2012.ppt.

Sentence Splitter, (5) Gazetteer(s), (6) English Tokenizer, (7) Document Reset PR.[‡] The resources can be accessed via either the GATE GUI, as in figure 3 (left), a detail of the upper frame of the Resources pane seen in figure 2, or the Command Line, shown in figure 3 (right). The lines in figure 3 link references to the same resource from different interfaces.

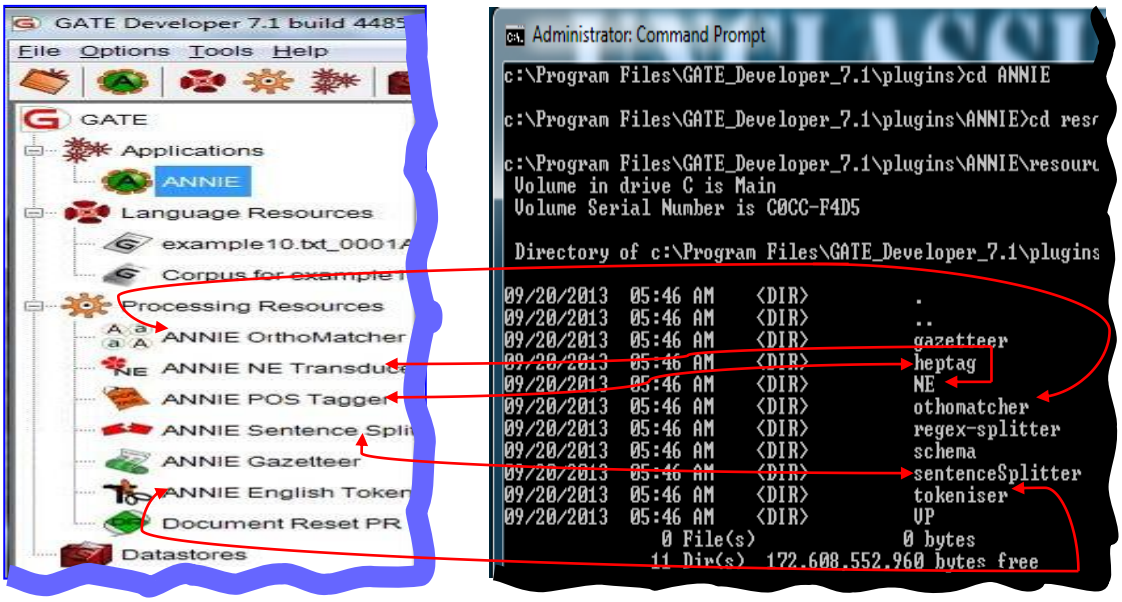


Figure 3. GUI and command line listings of ANNIE PRs.

3.2 Adapting Resources

Thus far in our work of developing SNEG-defined SYNCOIN truth data and a version of ANNIE customized for SYNCOIN data entities, we have mastered few steps. Nevertheless, we have discovered that these are quite robust and may be the sole adaptation techniques required for the task. For this pilot exercise in tagging SYNCOIN data, the resources needed and created consist of only Gazetteer lists and extensions using JAPE rules, effectively limiting GATE customization to a mere two of the seven available ANNIE PRs. In section 3.4, JAPE rule format and referencing framework are described. Future technical reports will examine the functionality of the JAPE language as well as that of the remaining five PRs.

For SYNCOIN data, use of the Gazetteer PR consists of simple table look-up. Satisfying SNEG for a GATE 7.1 installation involves modification of files, i.e., lists of names in a single person, location, or organization category. Editing entails (1) specifying new list items in files designated as person.lst, location.lst, and organization.lst and (2) updating the file named lists.def to point to these files. But listing all possible entities may be difficult and may even result in ambiguity (see section 3.3). In that case, JAPE rules, which will be explained in section 3.4, can be created for the grammar. But while the use of Gazetteer lists is simple, the use of JAPE grammar rules is complex. JAPE rules combine to create sets, or phases, in phase files, which, in main.jape,

combine into multiphases to create grammars. Phases, multiphases, and grammars are discussed further in section 3.4. Rules are one of GATE's trademark adaptable resources, called by the IE engine from within a grammatical system defined in main.jape.

After a GATE install, the gazetteers called from lists.def follow the path C:\Program Files\GATE_Developer_7.1\plugins\ANNIE\resources\gazetteer and subdirectory, C:\Program Files\GATE_Developer_7.1\plugins\ANNIE\resources \NE is the location of the pointer file, main.jape.*

3.3 Ambiguity

Lexical ambiguity is a common challenge for automatic language processing and one that is shared at the entity reference level by GATE ANNIE. An entity reference is an expression that points to a single entity, or set of thereof, outside the text. Ambiguity occurs when one expression string is an exact match with another string, with the latter constituting either an unrelated linguistic constituent or an entirely different reference. The second string, then, can point to an entirely "other" outside individual or collective entity. In computational terms, we can imagine two strings input to a <string compare> function returning 1, which an automatic understanding system sends in very different directions. For example, the expression "May" in the text, "Dr. May recommends morphine," references a person, while the same expression, occurring elsewhere in the same text, "It was in May 2010 that he departed Belgrade," references a time frame. Unless programmed to recognize context, list-based IE systems are ill-equipped to distinguish between expressions that look alike but refer differently.

3.4 JAPE Pattern Matching

ANNIE's Gazetteers—basically lists themselves—are designed to identify string matches only, to support tagging. It is beyond the scope of their design to resolve such ambiguity. It is for this reason, among others, that GATE is equipped with PRs for pattern matching as well as string matching. JAPE is the pattern-matching language for GATE. A JAPE rule has a <left-hand-side> (LHS) condition, input string match to ordered text pattern, and a <right-hand-side> (RHS) action that the system is programmed to take, when the condition is true.

As mentioned in section 3.2, JAPE grammar rules can be complex, and the file main.jape keeps track of these rules with a listing out of the names of files containing a phase or set of related rules. The rules are related because they treat a single linguistic category, which may manifest in different ways, each requiring its own rule. For example, time reference format can change depending on context, as with the formats, "two o'clock," "2:00," "2 p.m.," and "1400." Each rendering warrants its own rule. The four rules, each handling a distinct format, would then occupy space in the same file, named for the unifying linguistic category. In this case, if the phase file were named times.jape, this name would have to appear among the phase filenames

*Note that the terms "JAPE transducer," "Named Entity" or "NE Transducer," "NE Tagger," and "JAPE grammar" are equivalent references.

listed in main.jape for the patterns to be recognized and the instantiating strings to be automatically annotated. Recall that phase files, created to accommodate a set of complementary language categories in a corpus, constitute for GATE a grammar or multiphase. One is likely to find computational linguists creating specialized grammars for annotation of information relevant to specific domains, genres, registers, media, or combinations thereof.

Phases, considered in the abstract, are linguistic categories represented by sets of strings that match a pattern. Conceptualized concretely as files, phases serve to structure related JAPE rules to identify and tag instantiating strings. When JAPE text-pattern-based matching rules combine in a file to create a phase, the phase file, say times.jape, consists of, generally, with possible added information, a simple slot-filler template of the form

```
Phase: Phase Category Time

  Rule: Alphabetic-PCT-Type001
    Rule: NumberPunct-PCT-Type002
    Rule: AlphaNumeric-PCT-Type003
    Rule: Universal-PCT-Type004
                ...
    Rule: YetAnother-PCT-TypeNNN
```

Similarly, grammars or multiphases, when viewed abstractly, are sets of generally identifiable language category types associated with well-defined, linguistically or otherwise, corpora. In parallel fashion to the function of a phase file, a grammar, when conceptualized concretely as a file, serves to structure the set of phase files created for a given corpus or task. The modular organization permits, in serial runs of the IE engine, easy substitution of files, file groupings, and file grouping versions. This flexibility facilitates not only more fine-grained comparative analysis but also more complex and informative experimental design. When phase files combine to create a GATE grammar, the main.jape pointer file follows this, very general, example template:

```
MultiPhase: The Corpus-010 Grammar

 Phases:
   times.jape
   persons.jape
   locations.jape
   money.jape
```

A simple rule in the JAPE language appears as figure 4. It instructs IE tagging to recognize U.S. currency symbols, or dollar signs, "$." Imagine similar rules recognizing Mongolian, Korean, Ukrainian, or Thai currency symbols, i.e., tughrik, "₮," won, "₩," hryvnia, "₴," and baht, "฿," respectively.

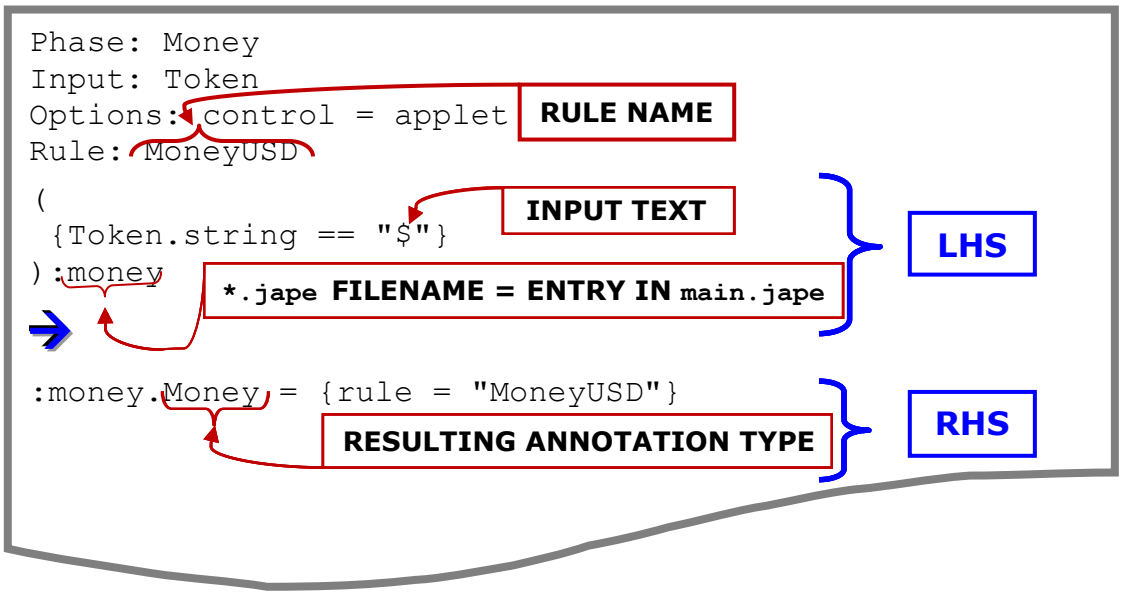


Figure 4. A simple JAPE grammar rule.

Note that LHS and RHS are separated by the arrow symbol "→" following standard JAPE rule formatting practices. JAPE files are generally named for the phase category they process. The phase "Money" here shares its name, in conventional fashion, with the JAPE file itself, the name of which occurs after the colon, as the final element in the rule's LHS. Entered before ANNIE runs into main.jape, filename money.jape passes control to the eponymous file for iterative rule-firing on single-symbol input, as defined by the <Token> type-filled <Input> slot on the file's second line.

The <Options> slot contains system details that vary depending on computing environment. Slot <Rule> fills with a rule name appropriate to a specific phase variant; thus rule <MoneyUSD> aptly references a "Money" phase rule for money type "U.S. Dollar." Similar names would similarly be applied to rules handling relevant types (e.g., <MoneyMONT> for type "Mongolian Tugrik," <MoneyKORW> for type "Korean Won," <MoneyUKRH> for "Ukrainian Hryvnia," and <MoneyTAIB> for "Thai Baht").

The LHS, in the figure, starts on the fifth line with an open parenthesis. A pair of braces on the sixth line contains the condition on which the action depends. In this case, that condition is that the input string must be a token and the token must be a dollar sign. After the closing parenthesis, there is a colon, indicating that the condition can be found in the file, the name of which follows the colon.

If the input fails to meet the condition, control passes to conditions in subsequent phase rules for testing. However, if the input meets the condition, control passes to the RHS, as indicated by the arrow, and the specific action described within the braces. In this case, the action is to set the variable rule to the value "MoneyUSD." Variable instantiation triggers—as indicated by the

equal sign—the action of annotation in the manner indicated. In this case, rule licenses input annotation with phase tag "Money," as defined by its rules in a file named "money." Phase tag names and filenames are indicated, respectively, by prefixes and suffixes of strings that follow initial RHS colons and precede triggering equal signs.

From the analyst's perspective, ANNIE is returning input text, now with category tags inserted around instantiating strings of the 'Money' phase category. Moreover, GATE is adding the relevant phase annotation type, causing it to appear among the Annotation Set types in the Annotation Pane of the output file display. See figure 2. Output can also be formatted such that begin and end tag positions are indicated in offset files (see section 3.5). JAPE files are placed in a GATE directory with relative path: ...\plugins\ANNIE\resources\NE.

Recall that both Gazetteer lists and JAPE grammar rules, along with their pointer files, are loaded automatically by GATE only if they appear in the previously referenced directories. Creating lists and rules is still an active area of development for preparing ground truth text data to support the evaluation of IE system performance on the SYNCOIN data. The intent here is to show why these resources are necessary to give unform resource locaters for the technology, and, thereby, to provide foundational background for ongoing and new implementations of GATE.

The JAPE overview in the 2012 University of Sheffield GATE training course and the line-by-line explication of JAPE rules, detailed in a 2009 tutorial, are excellent options for further study.[*]

3.5 Annotation Output

When conditions for the system to perform annotating actions are satisfied, the system annotates in one of two ways. It can perform in-line annotation, whereby the category tags assigned by the rule are inserted into the original text file around the matching input string, and it can perform offset annotation. If parameters for the latter approach are set, the system creates a file, which accompanies the analyzed material. Annotation is accomplished in three steps. The system (1) calculates the text file locations for a matching string's bounding characters, (2) associates them with the tag category assigned by the rule, and (3) registers the paired <<location references> and <tag category>> on a new line in the accompanying offset file, so called because, unlike the in-line mode in which tags are adjacent to their material, in offset mode, tags are set off or apart from the material. Offset mode permits preservation of the integrity of the original for runtime processing, downstream manipulation, or archival purposes.

GATE implements offset mode within the Datastores PR, which is called when a session of annotation is concluded and work is to be saved. An annotated corpus quickly becomes fairly large. Datastores permits loading of the unannotated corpus prior to processing, with negligible

[*]Module 3 of the course at https://gate.ac.uk/wiki/TrainingCourseJune2012/ introduces JAPE and can be found at http://gate.ac.uk/sale/talks/gate-course-jun12/track-1/module-3-jape/module-3-jape.pdf. A tutorial by D. Thakker, T. Osman, and P. Lakin from 2009 appears at https://gate.ac.uk/sale/thakker-jape-tutorial. There is also a JAPE repository at http://gate.ac.uk /jape-repository/.

overhead and space-saving storage of annotations as pointers from character positions within the corpus. Appendix B gives details for doing this. Automatic annotation for this project was done by GATE Developer 7.1 on a Windows 7 machine. At the start of the project in January 2013, 7.1 was the most recent version. The discussion should still be relevant if/when a later release becomes available

4. Machine Learning Resources

GATE also includes a plug-in for ML,* which approaches the annotation task in a manner quite different from that used in ANNIE. Rather than manually identify patterns and create rules for input matching, programs implementing ML techniques automatically uncover category-predicting patterns from minimally tagged text data or sets of seed elements. The predictive patterns are known as language models and created by a process of training from fairly large text corpora similar in domain, genre, and register.

When it comes to data, the concept of "enough" is relative; generally, the more the merrier. Factors such as structural complexity of extraction category, homogeneity of training material, as well as technique selected for experimentation affect this determination. One accepted practice is that in order to know how well ML models are performing, corpora are split into equal parts such that one part is set aside for testing while the rest are used to train the model. Serial selection of partitions with score averaging is called X-fold Cross Validation, X being the number of partitions created.

A design such as this permits a view into the effects of known data features, possibly spurious or idiosyncratic, present in one or more of the partitions. More important, by averaging scores obtained by models weak and strong, it also prevents anyone (e.g., one trained on corrupted data) from inaccurately characterizing the efficacy of the approach itself for the task.

ML automates the process of predicting where information category mentions occur in text with language models, which associates an information category with pattern-based locations where mentions have high likelihood of occurrence in existing data. In GATE, this means using manual annotation of different categories for training or seed data so that models reflect patterning of intercategory relations as well as likelihoods of occurrence.

Although currently state of the art in IE, ML techniques have yet to be tested for our SYNCOIN ground truth annotation effort. Once we have determined the effectiveness of our adaptation of the two resources described in this report, we can move on to experimentation using ML techniques. Precision, Recall, and F-Measure scores of SYNCOIN-adapted GATE engines on SYNCOIN data, covering various information categories, their definitions, and combinations

*GATE's ML plug-in operates from the directory Learning rather than Machine Learning; the latter is obsolete.

thereof, will constitute a baseline for follow-on ML experiments. This type of expanded experimentation is inevitable given that GATE includes the University of Waikato (New Zealand) Environment for Knowledge Analysis (WEKA), which consists of more than 150 algorithms for data mining tasks. GATE Developer has ML in the following directory: C:\Program Files\GATE_Developer_7.1\plugins\Learning.

5. Conclusions and Future Work

This report has introduced GATE, a versatile text annotation and IE environment, which comprises many more functionalities than those described here. GATE has subsumed open source tools as well as proprietary technologies and made them, be they PR or LR, available within its accessible interface. Routines are wrapped such that they can be selected for inclusion in a text processing pipeline. Thus, there are many more features of GATE to explore and with which to experiment, including remaining ANNIE resources, ML techniques, and incorporated community-developed technologies, among others.

The two featured ANNIE resources, the Gazetteers and the JAPE grammar language, have provided the SYNCOIN team a means for gauging the extent to which existing tools, applied to U.S. Army data, can be adapted to provide the metadata required for sophisticated real-time processing to support analyst and leadership decision-making, the overarching goal of this effort. We want also to extol the virtues of GATE Developer's Datastores facility. Despite a dearth of documentation, it achieved powerful economies of space, time, and complexity with offset pointers to tag locations in text.

In support of the SYNCOIN tagging project, we explored traditional annotation with GATE ANNIE, a fairly shallow, entity-centric, RB IE engine, and we intend to expand our purview with follow-on study of Ontology-Based Information Extraction (OBIE), also known as semantic annotation.

ANNIE uses a flat data structure, that is, information categories are characterized only by their text features and context, or their inclusion in a list, rather than their intra-class relationships. By contrast, semantic annotation uses a hierarchical or graphical data structure, which permits a richer representation, capable of expressing inter-class relationships in terms of structure and type. GATE's Ontology plug-in supports OBIE by leveraging such knowledge for understanding purposes.

These include, among others, (1) teasing apart similar meanings, as with learn/know, using lexical aspect features; (2) disambiguating unrelated concept/reference senses of homonyms or near-homonyms such as exact "precise"/exact "command" or mean "poor"/mean "ill-tempered" or still "unmoving"/still "up to present"/still "distiller" or even segments of name referents such

as Dr. in “Dr. Betty Fuchs”/“501 Winters Dr.” with one a title and the other a roadway or MD in “Marcus Dolby, MD”/“Chesapeake Beach, MD,” the former indicating a profession and the latter a geographical jurisdiction.

6. Bibliography

Chinchor, N. MUC-7 Named Entity Task Definition; version 3.5.;1997. http://www.itl.nist.gov /iaui /89.02/related _projects/muc/proceedings/ne_task.html (accessed 17 September 1997).

Cunningham, H.; Maynard, D.; Bontcheva, K.; Tablan, V.; Aswani, N.; Roberts, I.; Gorrell, G.; Funk, A.; Roberts, A.; Damljanovic, D.; Heitz, T.; Greenwood, M.; Saggion, H.; Petrak, J.; Li, Y.; Peters, W. Text Processing with GATE; Version 6. Department of Computer Science: University of Sheffield (UK), 15 April 2011; ISBN 0956599311.

Cunningham, H.; Bontcheva, K. Named Entity Recognition, GATE website. http://gate.ac.uk /sale/talks/ne-tutorial.ppt (accessed 10 May 2013).

Cunningham, H.; Maynard, D.; Bontcheva K.; Tablan V. GATE: An Architecture for Development of Robust HLT Applications. *Proceedings of 40th Annual Meeting of Association for Computational Linguistics (ACL),* Philadelphia, PA, July 2002; pp 168–175.

Cunningham, H.; Wilks, Y.; Gaizauskas, R. GATE – A General Architecture for Text Engineering. *Proceedings of the 16th Conference on Computational Linguistics (COLING-96),* Copenhagen, August 1996; pp 1057–1060.

GATE Embedded API Track 11, Module 5, Fifth GATE Training Course, Slide 7 of 61, University of Sheffield, UK, June 2012. https://gate.ac.uk/sale/talks/gate-course-may-10 /track-2/module-5-embedded/module-5-slides.pdf (accessed 15 July 2013).

GATE Website. http://gate.ac.uk/download/ (accessed 15 July 2013).

GATE Website. GATE Training Course Module 3: Introduction to JAPE, University of Sheffield, UK, June 2012. https://gate.ac.uk/wiki/TrainingCourseJune2012/; JAPE Website. http://gate.ac.uk/sale/talks/gate-course-jun12/track-1/module-3-jape/module-3-jape.pdf (accessed 1 May 2013).

Graham, J.; Hall, D.; Rimland, J. A Synthetic Dataset for Evaluating Soft and Hard Fusion Algorithms, 2011. SPIE Website. http://spiedigitallibrary.org/data/Conferences/SPIEP /62118/80620F_1.pdf (accessed 2 April 2013).

Graham, J.; Rimland, J.; Hall, D. SYNCOIN: A Synthetic Data Set for Evaluating Hard and Soft Fusion Systems. *Proceedings of the 14th International Conference on Information Fusion*, Chicago, IL, 2011.

Matuszek, P. CSC 9010: Text Mining Applications Fall 2012: Introduction to GATE. http://www.csc.villanova.edu/~matuszek/spring2012 /GATEOverview2012.ppt (accessed 18 May 2013).

Less Commonly Taught Languages Project Team, Simple Named Entity Guidelines, Less Commonly Taught Languages, v.6.5, 28 March 2006. Presented at Linguistic Data Consortium, Data Linguistic Consortium, Philadelphia, PA, 2006.

Thakker, D.; Osman, T.; Lakin, P. GATE JAPE Grammar Tutorial v. 1.0., 2009. GATE Website. https://gate.ac.uk/sale/thakker-jape-tutorial (accessed 11 March 2013).

INTENTIONALLY LEFT BLANK.

Appendix A. Running A Nearly New Information Extraction (ANNIE) Extraction on a File of 595 Concatenated SYNCOIN Messages

The following figure results from applying nearly new information extraction (ANNIE) in GATE Developer to all 595 SYNCOIN messages concatenated into in a single, very long document. Note that only the beginning of text output is displayed. Messages are separated by date in a <mm/dd/yy> format and highlighted in red; location(s) of entity(s) has blue highlights. Once GATE Developer is started, the steps for output include the following:

(1) Load ANNIE system with defaults from the tool bar.

(2) Right-click Language Resources: New -> GATE Document -> SYNCOINall.xml; open.

(3) Under Language Resources, right-click SYNCOINall.xml_00; New Corpus with this Document.

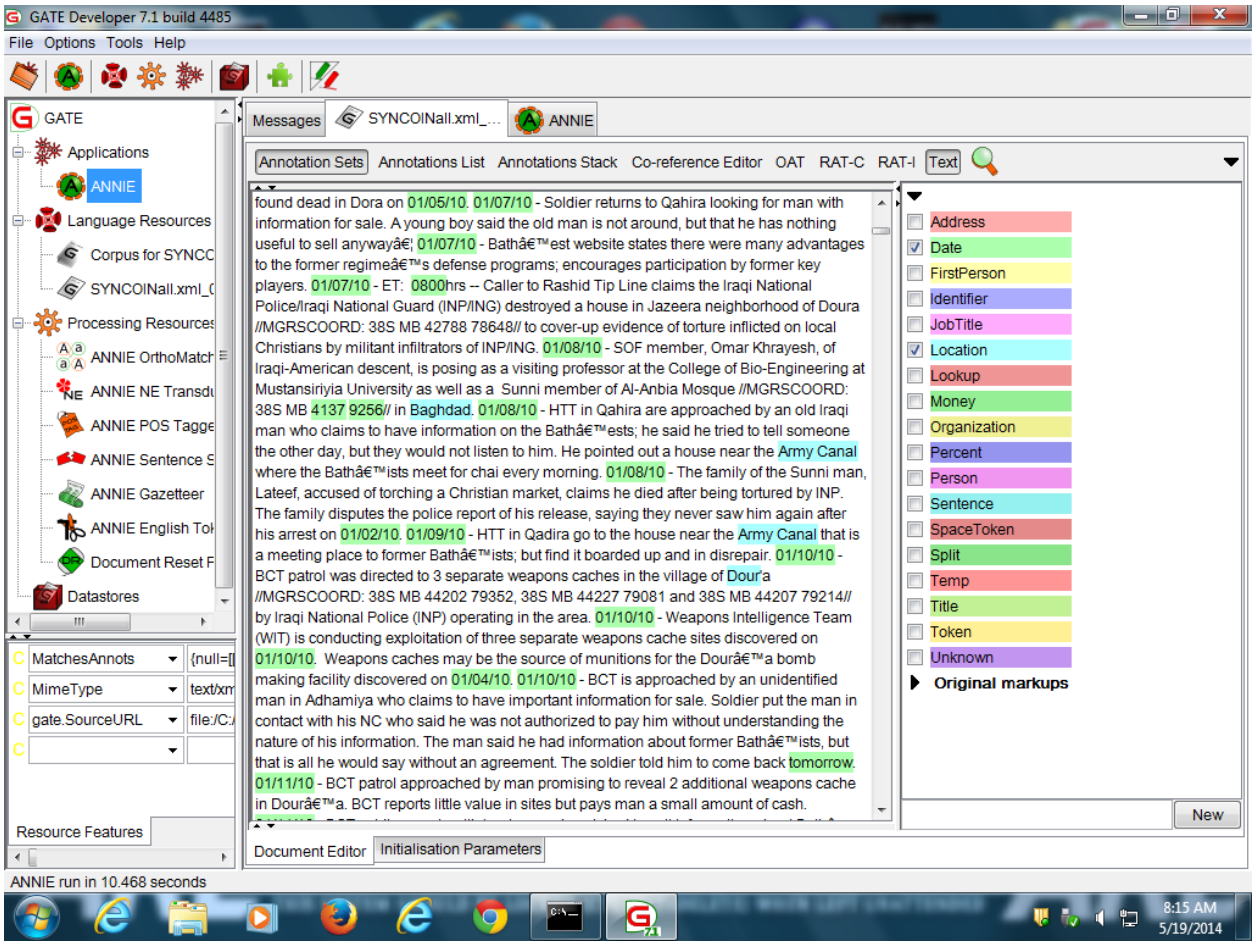
(4) Under Language Resources, double-click SYNCOINall.xml_000.

(5) Under Applications, double-click ANNIE. Run this application.

(6) Left-click on SYNCOINall.xml_000 tab.

(7) Left-click Annotation Sets tab.

(8) Check annotation type, e.g., Location.



Appendix B. Running A Nearly New Information Extraction (ANNIE) Extraction on a Corpus of SYNCOIN Messages

A General Architecture for Text Engineering (GATE) Datastore corpus can be used for showing individual messages. Taking the steps in the following example results in the formatting for display of 85 messages from the 595-message SYNCOIN corpus; the first is shown in the following figure. Date(s) is highlighted in red and location(s) in blue. Once GATE Developer 7.1 (or >) is started, the steps for output include:

(1) Right-Click Language Resources:

New => GATE Corpus => Name: SYNCOIN1-85_07012013

(2) Right-Click Datastores:

Create Datastore =>

SerialDataStore: Java-Serialized File-Based storage =>

SYNCOIN1-85DataStore07012013 => OPEN

(3) Language Resources:

Right-click SYNCOIN1-8507012013 =>

Populate => <name of directory with the 85 messages>

Open.

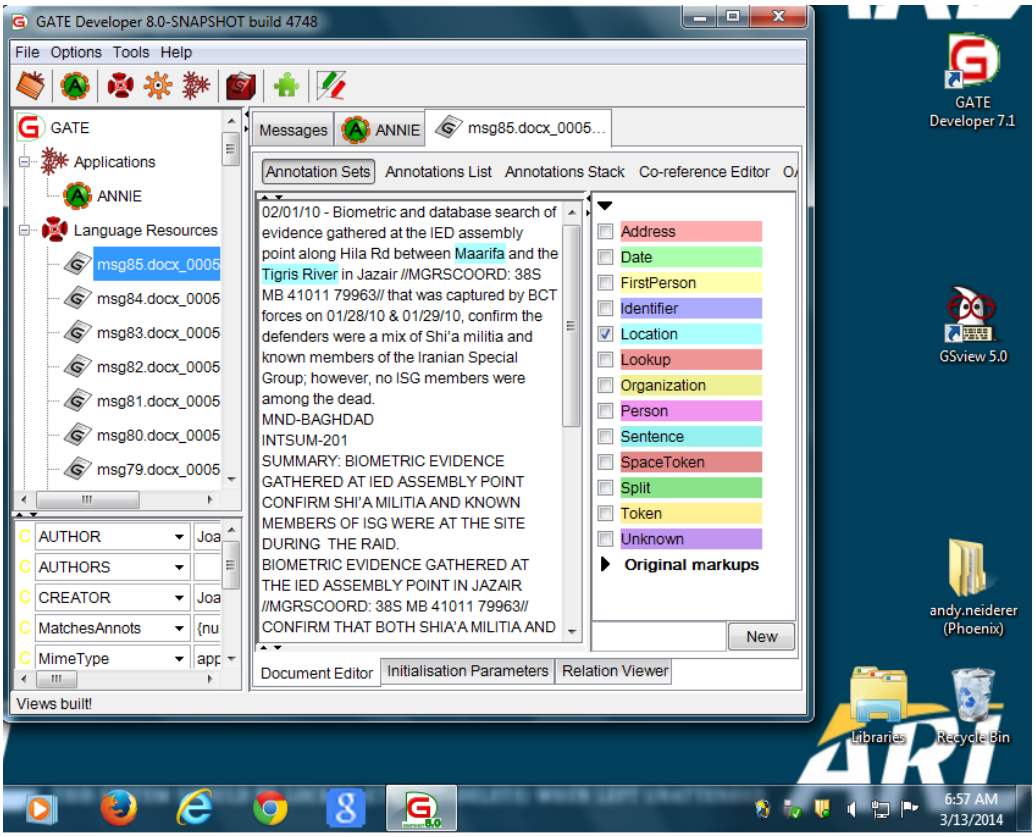
(4) Load nearly new information extraction (ANNIE) system with defaults from the tool bar.

(5) Under Applications, double-click ANNIE; run this application.

(6) Double-click on any of the 85 messages.

(7) Left-click Annotation Sets tab.

(8) Check annotation type, e.g., Location.


GATE Developer 8.0-SNAPSHOT build 4748
File Options Tools Help
GATE
Applications
ANNIE
Language Resources
msg85.docx_0005
msg84.docx_0005
msg83.docx_0005
msg82.docx_0005
msg81.docx_0005
msg80.docx_0005
msg79.docx_0005
Messages
ANNIE
msg85.docx_0005...
Annotation Sets
Annotations List
Annotations Stack
Co-reference Editor
02/01/10 - Biometric and database search of evidence gathered at the IED assembly point along Hila Rd between Maarifa and the Tigris River in Jazair //MGRSCOORD: 38S MB 41011 79963// that was captured by BCT forces on 01/28/10 & 01/29/10, confirm the defenders were a mix of Shi'a militia and known members of the Iranian Special Group; however, no ISG members were among the dead.
MND-BAGHDAD
INTSUM-201
SUMMARY: BIOMETRIC EVIDENCE GATHERED AT IED ASSEMBLY POINT CONFIRM SHI'A MILITIA AND KNOWN MEMBERS OF ISG WERE AT THE SITE DURING THE RAID.
BIOMETRIC EVIDENCE GATHERED AT THE IED ASSEMBLY POINT IN JAZAIR //MGRSCOORD: 38S MB 41011 79963// CONFIRM THAT BOTH SHIA'A MILITIA AND
Address
Date
FirstPerson
Identifier
Location
Lookup
Organization
Person
Sentence
SpaceToken
Split
Token
Unknown
Original markups
New
AUTHOR
AUTHORS
CREATOR
MatchesAnnots
MimeType
Document Editor
Initialisation Parameters
Relation Viewer
Views built!
GATE Developer 7.1
GSview 5.0
andy.neiderer (Phoenix)
Libraries
Recycle Bin
6:57 AM
3/13/2014

1 DEFENSE TECHNICAL
(PDF) INFORMATION CTR
DTIC OCA

2 DIRECTOR
(PDF) US ARMY RESEARCH LAB
RDRL CIO LL
IMAL HRA MAIL & RECORDS MGMT

1 GOVT PRINTG OFC
(PDF) A MALHOTRA

2 DIR USARL
(PDF) RDRL CII C
A NEIDERER
RDRL CII T
M VANNI